cdtime – time utilities module

The “cdtime” module

- CDAT uses its own ***cdtime*** module to manage time and temporal coordinate reference systems (or calendars).

- cdtime is integral to CDMS, but also very useful tool for users interacting with data.

The “cdtime” module

cdtime provides useful functionality to:

- create time objects attached to datasets.
- refer to different calendar types (such as Gregorian and 360-day year calendars).
- convert between an absolute and relative time description.
- work with time intervals.

Component and Relative Time

Two time descriptions available:

1. Component Time:

– Integer fields for (year, month, day…second), e.g:

```
>>> ct=cdtime.comptime(1999, 6, 12, 18)
>>> print ct
1999-6-12 18:0:0.0
```

2. Relative Time:

– A floating point value and a relative units string (and base time), e.g.:

```
>>> rt=cdtime.reltime(11.75, \
      “days since 1999-06-01 00:00”) # same as above
>>> print rt
11.75 days since 1999-06-01 00:00
```

Why Component and Relative Time?

- The two types provide helpful functionality and versatility when representing time objects.

- A ‘comptime’ or ‘reltime’ object is returned.

The standard time units definition

- But first, let's get familiar with NetCDF-style time definitions…

- cdtime and CDMS time axes follows the NetCDF convention for representing time. Relative time is time relative to a fixed base time. It consists of:

  – a **units** string, of the form **"units since basetime"**, and a floating-point **value.**

Time units from UDUNITS

The specification [from UDUNITS]:

“seconds since 1992-10-8 15:15:42.5 -6:00”

- indicates seconds since October 8th, 1992 at 3 hours, 15 minutes and 42.5 seconds in the afternoon in the time zone which is six hours to the west of Coordinated Universal Time (i.e. Mountain Daylight Time).

- The time zone specification can also be written without a colon using one or two-digits (indicating hours) or three or four digits (indicating hours and minutes).

cdtime examples (1)

So how do you use cdtime? Here are some examples:

- You know a time interval and the units but no absolute (component) time:

```
>>> ct=cdtime.reltime(12, "hours since 1981-1-1")
>>> print ct
12.00 hours since 1981-1-1
```

- You would like to add 1.5 days to a cdtime object:

```
>>> newtime=ct.add(1.5, cdtime.Days)
>>> print newtime
48.00 hours since 1981-1-1
```

cdtime examples (2)

- You have a dataset with a known reference time and you are adding a new value with a known absolute (component) time :

```
>>> ct=cdtime.comptime(1999,6,12,18)
>>> print ct
1999-6-12 18:0:0.0
>>> ct2rt=ct.torel("days since 1966-12-31 06:00")
>>> print ct2rt
11851.50 days since 1966-12-31 06:00
```

- To compare two time objects:

```
>>> if ct1.cmp(ct2): # ct1 and ct2 pre-defined
...     print “They are the same!”
“They are the same!”
```

Calendars in cdtime

- A calendar specifies the number of days in each month, for a given year. **cdtime** supports the following:
  - Gregorian
  - Mixed (Julian/Gregorian) = Default
  - Julian
  - NoLeap (Year)
  - Calendar360 (all days have 30 months).

- Typical usage is when you are converting or modifying cdtime objects:

```
>>> newtime=ct1.add(25, cdtime.Hours, \
            cdtime.Calendar360)

>>> newtime2=relt.tocomp(cdtime.GregorianCalendar)
```